NAND 플래시 메모리 기반 SSD 에서의 블록 파편화 문제에 관한 연구[1)]

하병민[O*], 조현진[*], 엄영익[*]
[*]성균관대학교 정보통신공학부
e-mail:{chaosken, hjcho, yieom}@ece.skku.ac.kr

A Study on the Block Fragmentation Problem of SSD based on NAND Flash Memory

Byungmin Ha[O*], Hyunjin Cho[*], Young Ik Eom[*]
[*]School of Information and Communication Engineering,
Sungkyunkwan University

요 약

최근 낸드 플래시 메모리를 이용한 SSD(Solid State Disk)는 차세대 저장 매체로서 주목받고 있다. SSD의 성능을 높이기 위해 다양한 기술이 연구되고 있지만, 그 중 병렬성 향상 기법이 성능에 가장 큰 영향을 미치고 있다. 이에 따라 SSD에서는 병렬성 향상을 위해 논리 주소 스트라이핑(Striping)을 사용하고 있다. 논리 주소 스트라이핑을 이용하면 병렬성이 크게 증가되지만 매핑 기법에 따라 블록 활용률에 문제점이 발생할 수 있다. 본 논문에서는 논리 주소 스트라이핑 기법과 하이브리드 매핑 기법이 동시에 SSD에 적용되었을 때 발생하는 블록 파편화 문제를 확인하고, 블록 파편화 문제가 블록 활용률과 성능에 미치는 영향을 분석하였다.

1. 서 론

낸드 플래시 메모리는 비휘발성 메모리로서 소형화에 유리하고, 충격에 강한 장점으로 널리 사용되고 있다. 낸드 플래시 메모리를 이용한 SSD(Solid State Disk)는 저전력, 고성능, 고신뢰성의 특징을 바탕으로 기존의 저장매체인 하드 디스크를 대체할 매체로 각광받고 있다 [1].

그러나 낸드 플래시 메모리는 읽기, 쓰기 및 지우기 연산값이 다른 특징을 가지고 있으며, 지우기 전 기록할 수 없다는 제약조건(erase-before-write)을 가지고 있다 [2]. 이러한 특징을 숨기기 위하여 플래시 전환 계층(Flash Translation Layer)이라는 시스템 소프트웨어가 필요하다. 플래시 전환 계층은 운영체제와 낸드 플래시 메모리 사이에 위치하여 운영체제로부터 요청된 주소를 낸드 플래시 메모리내 물리 주소와 매핑하는 역할을 수행한다. 주소 매핑 기법은 페이지 매핑(page-level mapping), 블록 매핑(block-level mapping), 하이브리드 매핑(hybrid mapping)의 3가지로 구분할 수 있다 [3]. 페이지 매핑 기법은 낸드 플래시 메모리의 페이지 단위로 매핑 정보를 구성하며, 가장 좋은 성능을 보이지만 매핑 테이블의 크기가 매우 크다는 단점을 가진다. 블록 매핑 기법은 블록 단위로 매핑 정보를 구성하며, 페이지 매핑 기법에 비해 매우 적은 매핑 테이블 크기를 가지지만 데이터 갱신을 위한 오버헤드가 커진다는 단점이 있다. 하이브리드 매핑은 페이지 매핑 기법과 블록 매핑 기법을 혼합한 기법이며, 낸드 플래시 메모리의 일정 공간을 갱신 데이터를 전담하는 로그 버퍼(log buffer)로 활용하게 된다. 따라서 하이브리드 매핑 기법에서는 데이터 블록 영역과 로그 블록 영역으로 구성되며, 데이터 블록은 블록 매핑 기법으로, 로그 블록은 페이지 매핑 기법으로 매핑 정보를 관리한다. 현재 가장 많이 활용되는 매핑 기법은 하이브리드 매핑 기법이며, 이에 따라 많은 연구가 진행되었다 [4-7].

SSD는 내부에 다수의 낸드 플래시 메모리를 병렬적으로 구성하며, 성능 향상을 위해 다양한 병렬성(parallelism) 향상 기법을 사용한다. 이 중 대표적인 것이 논리 주소 스트라이핑(striping) 기법이다. 논리 주소 스트라이핑 기법은 SSD에 할당되는 논리 주소 영역을 전체 낸드 플래시 메모리에 분산시켜 I/O 작업 수행시 최대한 많은 수의 낸드 플래시 메모리가 동작하도록 만드는 기법이다. 완전 스트라이핑(full striping) 기법으로 논리 주소 영역을 지정하게 되면, SSD 내부를 구성하는 낸드 플래시 메모리 전체를 단일 논리 주소 영역으로 구성하고 이에 따라 데이터 기록 범위가 전 영역에 펼쳐지게 된다 [8-9]. 이때 하이브리드 매핑 방식을 사용하여 데이터를 기록할 경우 블록에서 데이터가 기록될 페이지

1) 본 과제(결과물)는 교육과학기술부·지식경제부의 출연금으로 수행한 산학협력중심대학육성사업의 연구결과입니다.

위치가 고정된다. 그러나 낸드 플래시 메모리는 블록에 데이터를 기록할 때 첫 번째 페이지부터 순서대로 기록해야 하는 제약 조건을 가지고 있다 [2]. 때문에 아무런 데이터가 기록되어 있지 않은 공간을 활용할 수 없는 파편화(fragmentation) 현상이 발생하게 된다. 빈번한 블록 파편화 현상은 SSD의 저장 공간 낭비를 초래하고 이는 더 잦은 병합 연산을 발생시킨다.

본 논문에서는 SSD에서 발생하는 파편화 현상을 분석하고, 다양한 작업 패턴의 블록 활용률을 측정한다. 또한 파편화 현상이 SSD 성능 감소에 미치는 영향을 분석한다. 본 논문의 구성은 다음과 같다. 2장에서는 낸드 플래시 및 SSD 내부 구조를 설명하고 3장에서는 앞에서 언급한 파편화 문제점을 정의한다. 4장에서는 실험 결과를 보이며 5장에서 본 논문의 결론을 맺는다.

2. 관련연구

단일 낸드 플래시 메모리는 제조회사 및 종류에 따라 4GB 또는 8GB의 크기를 가진다. 그림 1에서는 낸드 플래시 메모리의 일반적인 구조를 보인다.

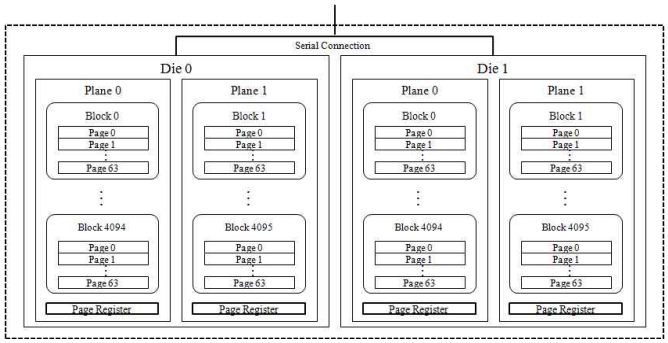

그림 1. 플래시 메모리 내부 구조

그림 1에서 보이는 낸드 플래시 메모리의 경우 크기는 4GB이며 내부는 다이 (die), 플레인 (plane), 블록 (block), 페이지 (page)로 구성된다. 또한 물리적으로 구분된 2개의 다이를 이용하여 내부의 병렬성을 높인다. 각 다이들은 데이터 전송 버스를 공유하고, 병렬적인 I/O를 수행할 수 있다. 또한 다이는 플레인들을 포함하며 플레인은 다수의 블록들로, 블록은 다수의 페이지들로 구성된다.

I/O 수행시 운영체제는 해당 데이터를 논리 블록 주소 (LBA: Logical Block Address)를 통해 접근하게 된다. SSD 내부의 FTL은 SSD 내부의 페이지 크기에 따라 다수의 LBA를 하나의 논리 페이지 번호 (LPN: Logical Page Number)로 변환 한다. 만약 LBA의 단위가 512 bytes이고 SSD의 페이지가 2KB라면 4개의 LBA를 하나의 페이지로 구성하게 된다. SSD의 주소 정보를 변환하기 위해 페이지 매핑 기법을 사용할 경우 데이터 기록 요청이 발생하면 빈 페이지에 데이터를 기록한 후 해당 논리 페이지 번호와 물리 페이지 번호 (PPN: Physical Page Number)와의 매핑 정보를 유지한다. 만약 하이브리드 매핑 기법을 사용한다면 LPN과 PPN의 매핑 정보 대신 데이터 블록은 블록 매핑 정보를, 로그 블록은 페이지 매핑 정보를 유지하게 된다. 이 때 LPN을 이용하여 논리 블록 번호 (LBN: Logical Block Number)와 페이지 오프셋을 계산하고 처음 기록 요청이 발생된 데이터라면 데이터 불록에 기록한 후 LBN과 물리 블록 번호 (PBN: Physical Block Number)의 매핑 정보를 유지한다. 이후 블록 내 데이터 갱신이 발생하면 블록에 기록되어 있는 이전 데이터를 무효화 (invalidate)한 후 로그 블록을 할당하여 갱신 데이터를 기록한다. 그러나 앞에서 살펴본 바와 같이 SSD 내부의 LPN은 물리 페이지 위치에 상관없이 고정시켜 사용하기 때문에 LBA를 이용하여 LPN을 계산할 수 있으며 따라서 LBA와 LPN의 매핑 정보는 유지할 필요가 없다. 결국 LPN과 PPN 또는 LBN과 PBN과의 정보만 유지하면 된다. SSD는 내부의 낸드 플래시 메모리들을 최대한 병렬적으로 동작시키기 위해 LPN을 SSD 전 영역에 스트라이핑 (striping) 한다. 그림 2에서는 스트라이핑 기법사용 유무에 따른 LPN 할당 차이를 보인다.

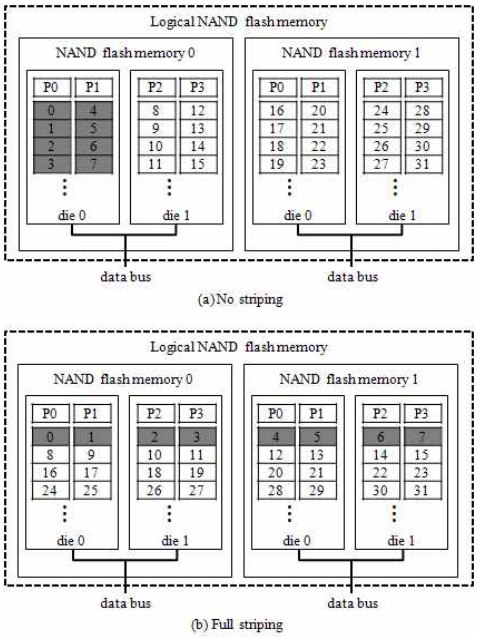

그림 2. SSD 내부 논리 주소 할당

그림 2에서 보이는 것처럼 SSD는 내부에 가지고 있는 모든 물리 플래시 메모리를 하나의 논리 플래시 메모리로 구성하며, 낸드 플래시 메모리에 포함된 다이와 플레인의 개수는 각각 2개라고 가정한다. 그림 2에서 데이터 기록 요청이 발생하여 LPN 0~7까지 8개의 페이지에 데이터를 기록해야 한다고 가정한다. 그림 2-(a)처럼 LPN을 스트라이핑 하지 않는다면 8개의 페이지를 기록할 때 병렬성을 활용할 수 없게 된다. 즉, 0번 낸드 플래시 메모리의 0번 다이에만 작업이 집중된다. 하지만 그림 2-(b)처럼 낸드 플래시 메모리 전 영역에 걸쳐 LPN을 스트라이핑 한다면 낸드 플래시 메모리간 그리고 다이간 병렬성을 활용할 수 있게 된다. 이때는 4개의 다이가 동시에 동작하기 때문에 no striping에 비해 이론적으로 기록 속도를 4배 증가시킬 수 있다 [8].

3. 블록 파편화(fragmentation) 문제

낸드 플래시 메모리는 자체적으로 데이터를 기록할 때 해당 블록의 첫 번째 페이지부터 순차적으로 기록해야 하는 제약 조건이 있다 [그림 3].

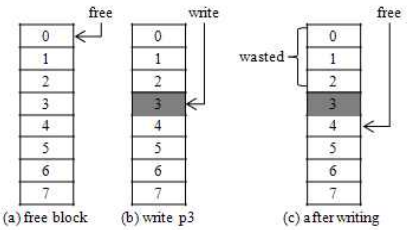

그림 3. 낸드 플래시 메모리의 제약 조건

그림 3에서 하나의 블록은 8개의 페이지로 구성된다고 가정한다. 그림 3-(b)처럼 데이터를 빈 블록의 p3에 기록한다면, 다음 번 기록 가능한 빈 페이지는 p4가 된다 [그림 3-(c)]. 이 때 해당 블록의 p0~p2에는 빈 페이지 임에도 불구하고 데이터를 기록할 수 없게 된다. 따라서 이 경우 해당 블록은 3개의 페이지 공간을 낭비하게 된다. 이때 데이터의 블록 내 페이지 오프셋이 고정되지 않은 페이지 매핑 기법을 사용하게 되면 위에서 언급한 문제는 발생하지 않지만, 블록 매핑 또는 하이브리드 매핑 기법을 사용하게 되면 위 문제점이 발생한다. 이에 따라 데이터 블록의 활용률 (utilization)이 낮아질 수 있다. 더구나 SSD 내부 병렬성을 높이기 위해 논리 주소 영역을 완전 스트라이핑 하게 되는데, 이 경우에는 문제점이 더 커질 수 있다. [그림 4]

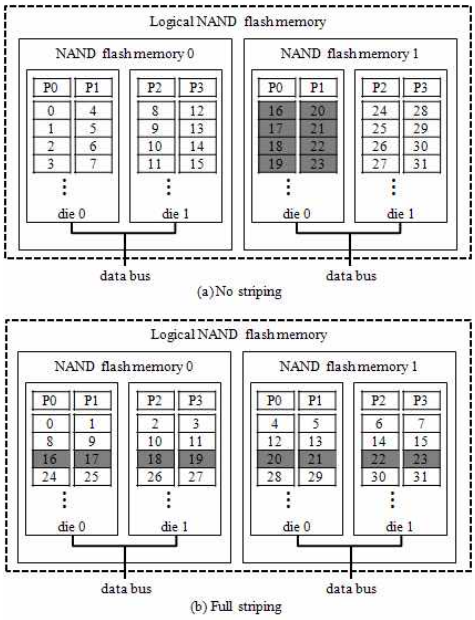

그림 4. Full striping을 사용할 경우 발생가능한 문제점

그림 4처럼 p16~p23까지 8개 페이지에 대한 기록 요청이 발생했다고 가정한다. No striping의 경우 그림 4-(a)와 같이 1번 플래시 메모리의 0번 다이에 있는 첫 번째 페이지부터 데이터가 기록된다. 하지만 그림 4-(b)와 같이 완전 스트라이핑의 경우 SSD를 구성하고 있는 전체 영역에 걸쳐 데이터가 기록되기 때문에 앞에서 언급한 낸드 플래시 메모리의 제약 사항에 따라 p0~p15가 빈 공간임에도 불구하고 해당 블록이 지워지기 전까지 데이터를 기록할 수 없게 된다.

4. 블록 파편화 문제 분석

앞에서 언급한 블록 파편화 문제점을 확인하기 위해 SSD 시뮬레이터를 이용하여 성능 측정을 수행하였다. 해당 시뮬레이터는 SSDsim [8] 기반으로 구현 하였으며, 페이지 매핑과 하이브리드 매핑 기법을 적용하여 실험하였다. 실험에 적용한 하이브리드 매핑 기법은 FAST [5] 기법이며, 로그 블록의 개수는 2,048 개로 설정하였다. 또한 IOzone [10] 벤치마크와 동영상, 웹서핑, 음악 재생 등 실제 PC 작업 환경을 기록한 PC application 작업 부하 (I/O trace)를 이용하여 실험하였다. 표 1에서 실험에 사용한 작업 부하의 특징을 보인다.

표 1. 실험에 사용한 작업 부하의 특징

이름	총 작업량	평균 요청 크기
IOzone	3.42 GB	58.68 KB
PC application	1.62 GB	13.53 KB

표 1에서 보이는 것처럼 IOzone 작업 부하는 한번에 큰 크기의 작업을 요청하며, 순차 패턴을 가지고 있다. PC application 작업 부하는 한번에 작은 크기의 작업을 요청하며, 랜덤과 순차 패턴을 동시에 포함하고 있다. 그림 5에서 각 작업 부하가 디스크에 접근하는 패턴을 보인다.

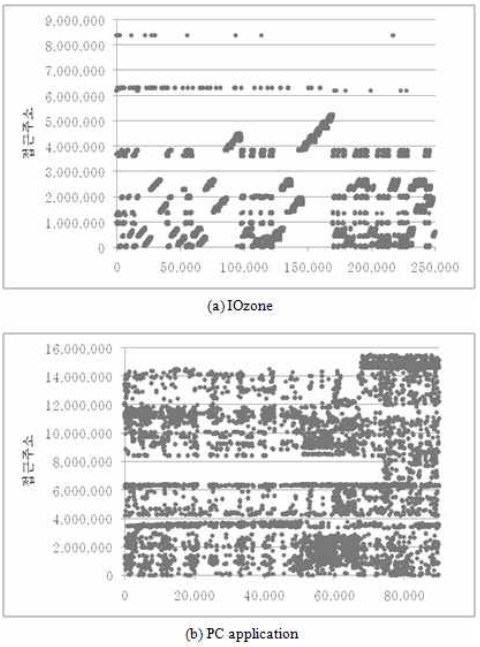

그림 5. 작업 부하의 디스크 접근 패턴

그림 5-(a)에서 보이는 것처럼 IOzone 작업 부하는 디스크에 순차적으로 접근하며 공간적 지역성이 큰 것을 확인할 수 있다. 반면 PC application 작업 부하는 그림 5-(b)와 같이 디스크의 전 영역에 작은 크기의 작업을 요청하는 패턴임을 확인할 수 있다.

그림 6에서는 페이지 매핑과 하이브리드 매핑의 블록 활용률 비교를 보인다. 데이터가 기록될 페이지 오프셋이 정해지지 않은 페이지 매핑 기법에서는 IOzone 작업 부하와 PC application 작업부하 모두 블록 활용률이 100%로 모든 페이지를 사용하고 있지만, 하이브리드 매핑 기법에서는 랜덤 특성을 가지고 있는 PC application 에서 매우 낮은 블록 활용률을 보이고 있다. 반면 순차 특성을 가진 IOzone 작업 부하는 높은 블록 활용률을 보이고 있다.

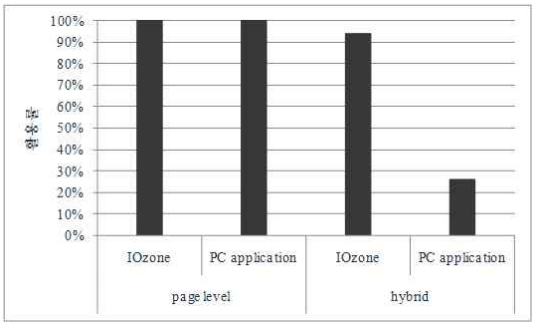

그림 6. 블록 활용률

그림 7에서는 페이지 매핑과 하이브리드 매핑의 비교를 보인다. 페이지 매핑의 경우 데이터 블록과 로그 블록의 구분이 없지만 하이브리드 매핑의 경우 데이터가 기록되는 위치는 데이터 블록과 로그 블록으로 구분된다. 데이터가 처음 데이터 블록에 기록 (*write_data*)된 비율과, 그 후 갱신이 발생하여 로그 블록에 기록될 때의 경우를 측정하였다. 또한 로그 블록에 기록될 경우를 다시 두 가지로 구분하였다. 첫 번째는 데이터의 순수한 갱신으로 인한 기록 (*update_pure*)이고 두 번째는 데이터 블록의 페이지가 빈 페이지임에도 불구하고 낸드 플래시 메모리의 제약사항으로 인해 로그 블록에 기록 (*update_wasted*)되는 경우이다. 실험을 통해 전체 데이터를 기록할 때 수행하는 *write_data*, *update_pure*, *update_wasted*의 비율을 측정하였다.

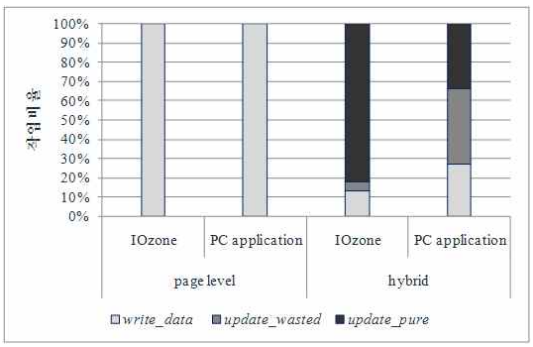

그림 7. 페이지 매핑 기법과 하이브리드 매핑 기법 비교

그림 7의 페이지 매핑의 경우 로그 블록의 개념이 존재하지 않기 때문에 처음 기록되는 데이터와 갱신 데이터의 구분 없이 모든 기록은 *write_data* 가 된다. 하지만 하이브리드 매핑을 사용할 때 갱신 데이터는 로그 블록에 기록해야 하는데, 이 때 작업 부하의 주소 접근 패턴에 따라 *update_pure*와 *update_wasted*의 비율이 달라진다. IOzone 작업 부하의 경우 공간적 지역성이 매우 크고 순차적인 패턴을 가지고 있기 때문에 데이터 블록에 순서대로 기록하게 되며 이에 따라 *write_data*와 *update_pure*가 많이 발생한다. 하지만 PC application 작업 부하의 경우 랜덤한 주소 접근 패턴을 많이 포함하고 있다. 따라서 처음 데이터 블록에 데이터를 기록할 때 첫 번째 페이지부터 기록하지 못하고 중간에 파편화된 공간을 만들게 된다. 이에 따라 다음 데이터 기록 시 블록 내 빈 페이지가 존재함에도 불구하고 갱신 데이터로 간주되는 경우가 발생한다. 이로 인해 IOzone 작업 부하에 비해 *update_wasted*가 매우 큰 비율을 차지하고 있다.

즉, 페이지 매핑 기법에서는 블록 파편화 문제가 발생하지 않고 하이브리드 매핑 기법의 데이터 블록 영역에서 블록 파편화 문제가 발생하고 있다. 블록 파편화 문제로 인해 발생하는 *update_wasted*가 많아지면 로그 블록에 기록되는 데이터가 많아진다. 따라서 블록 파편화 문제가 없을 때에 비해 로그 블록을 더 빨리 소모하게 되며 그로 인해 낸드 플래시 메모리에서 비용이 가장 큰 병합 연산이 더 자주 발생하게 된다. 따라서 *update_wasted* 의 비율이 높을수록 SSD의 성능은 감소한다.

*update_wasted*가 SSD의 성능에 미치는 영향을 측정하기 위해, 낸드 플래시 메모리에서 블록 내의 페이지를 순서대로 사용해야 한다는 제약 조건이 없을 때 (*no_wasted*)와 제약 조건이 있을 때 (*wasted*)의 처리량을 측정하였다. 그림 8에서 *update_wasted*가 처리량에 미치는 영향을 보인다.

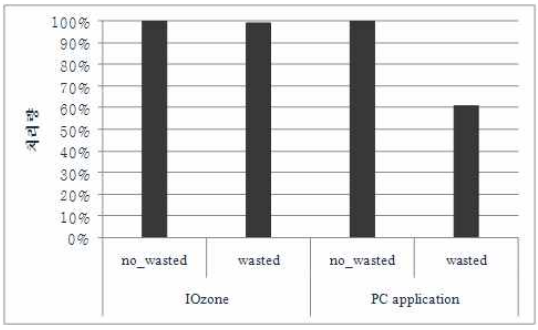

그림 8. *update_wasted* 가 처리량에 미치는 영향

*update_wasted*의 비율이 적은 IOzone 작업 부하는 *no_wasted*와 *wasted*의 성능 차이가 거의 없다. 하지만 *update_wasted*의 비율이 약 40%에 달하는 PC application 작업 부하의 경우 처리량이 40% 감소하였다.

5. 결론

SSD는 성능 향상을 위해 다수의 낸드 플래시 메모리를 병렬적으로 구성하며, 논리 주소 스트라이핑 기술을 이용하여 병렬성을 높인다. 하이브리드 매핑 기법은 FTL의 매핑 기법으로 가장 많이 활용되는 매핑 기법이며, 이에 따라 활발한 연구가 진행되었다. 하지만 하이브리드 매핑 기법은 데이터 블록 영역에서 블록 파편화를 발생시키는 문제점을 가지고 있으며, 블록 파편화는 SSD의 성능에 직접적인 영향을 미친다.

본 논문에서는 다수의 낸드 플래시 메모리를 이용하여 구성된 SSD에서 하이브리드 매핑 기법을 사용 하였을 때 블록 파편화 현상이 발생함을 보였다. 또한 랜덤 특성을 가진 작업 부하에서 블록 파편화로 인해 낮은 블록 활용률을 보임을 측정하였고, 이로 인해 SSD의 전체 성능이 감소함을 보였다.

참고문헌

[1] T. Dinkelman, "SSDs A Shift in Data Storage," in Proc. of Flash Memory Summit, 2008.

[2] Samsung Corporation. K9XXG08XXM Flash Memory Specification. http://www.samsung.com.

[3] T.-S. Chung, D.-J. Park, S. Park, D.-H. Lee, S.-W. Lee and H.-J. Song, "System Software for Flash Memory: A Survey," in proc. of Int'l Conf. of Embedded and Ubiquitous Computing (EUC'06), 2006.

[4] J. Kim, J. M. Kim, S. H. Noh, S. L. Min and Y. Cho, "A space efficient flash translation layer for compact flash systems," IEEE Transactions on Consumer Electronics, 2002.

[5] S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S. Park and H.-J. Song, "A log buffer-based flash translation layer using fully-associative sector translation," ACM Transactions on Embedded Computing Systems, 2007.

[6] C. Park, W. Cheon, J. Kang, K. Roh, W. Cho and J.-S. Kim, "A reconfigurable FTL (Flash Translation Layer) architecture for NAND flash based applications," ACM Transactions on Embedded Computing Systems, 2008.

[7] H.-J. Cho, D. Shin and Y. I. Eom, "KAST: K-Associative Sector Translation for NAND Flash Memory in Real-Time Systems," in proc. of Design, Automation and Test in Europe (DATE'09), 2009.

[8] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse and R. Panigrahy, "Design Tradeoffs for SSD Performance," in Proc. of USENIX Technical Conference, 2008.

[9] J-Y. Shin, Z-L. Xia, N-Y. Xu, R. Gao, X-F. Cai, S. Maeng and F-H. Hsu, "FTL Design Exploration in Reconfigurable High-Performance SSD for Server Applications," Int'l Conf. on Supercomputing (ICS'09), 2009.

[10] IOzone filesystem benchmark, http://www.iozone.org/.